1

REPORT DOCU

AD-A236 001

	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Feb. 1991	Final Scientific Report; 11/10/88-12/31/9

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Deductive Programming Synthesis	C: N00039-84-C-0211 T: 21, 25
6. AUTHOR(S) Zohar Manna	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Computer Science Department Stanford University Stanford, CA 94305	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
sponsoring agency: SPAWAR 3241C2 Space & Naval Warfare Systems Command Washington, D.C. 20363-5100	monitoring agency: ONR Resident Representative Mr. Paul Biddle Stanford Univ., 202 McCullough Stanford, CA 94305	

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release: distribution unlimited	

13. ABSTRACT *(Maximum 200 words)*

See page 1 of report.

Accession For	
NTIS GRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

14. SUBJECT TERMS	15. NUMBER OF PAGES
	11
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UL	UL	UL	UL

FINAL TECHNICAL REPORT

Sponsored by
Defense Advanced Research Projects Agency (DoD)
1400 Wilson Boulevard
Arlington, Virginia 22209-2389

"Deductive Programming Synthesis"
DARPA Order No. 6177
Issued by Space and Naval Warfare Systems Command
Under Contract # N00039-84-C-0211
Task 21: 11/10/88 - 2/28/90
Task 25: 3/1/90 - 12/31/90

Zohar Manna, Professor
Computer Science Department
Stanford University
Stanford, California 94305-2140

91-1662

FINAL TECHNICAL REPORT

Sponsored by
Defense Advanced Research Projects Agency (DoD)
1400 Wilson Boulevard
Arlington, Virginia 22209-2389

"Deductive Programming Synthesis"
DARPA Order No. 6177
Issued by Space and Naval Warfare Systems Command
Under Contract # N00039-84-C-0211
Task 21: 11/10/88 - 2/28/90
Task 25: 3/1/90 - 12/31/90

Zohar Manna, Professor
Computer Science Department
Stanford University
Stanford, California 94305-2140

91-1662

SUMMARY OF COMPLETED PROJECT

Program synthesis is the systematic derivation of a computer program to meet a given specification. The specification is a general description of the purpose of the desired program, while the program is a detailed description of a method for achieving that purpose.

The method is based on a deductive approach, in which the problem of deriving a program is regarded as one of proving a mathematical theorem. The theorem expresses the existence of an object meeting the specified conditions. The proof is restricted to be sufficiently constructive to indicate a method for finding the desired output. That method becomes the basis for a program, which is extracted from the proof.

The emphasis of the work has been on automating as much as possible of the program derivation process. Theorem-proving methods particularly well-suited to the program synthesis application have been developed. An interactive program-derivation system has been implemented. Applications to database management and planning have been investigated.

* * *

Reactive programs are programs whose role is to maintain an on-going interaction with their environment, rather than to produce a computational final result on termination. Programs belonging to this class are usually described by some of the attributes: concurrent, distributed, real-time, and typical examples of such programs are: embedded programs, communication networks, control programs of nindustrial plants, operating systems, real-time programs, etc. Clearly, the correct and reliable construction of such programs is one of the most challenging programming tasks existing today.

Due to the special character of these programs, the formal approach to their specification and development must be based on the study of their *behavior*, rather than on the function or relation they compute, an approach which is adequate for computational and sequential programs.

Formal methods were developed for the specification, verification and development of reactive programs, based on the formalism of *temporal logic* that has been specifically designed to reason about behaviors of reactive programs.

TECHNICAL INFORMATION

REACTIVE SYSTEMS

- **Temporal Proof Methodology for Reactive Systems** ([MP2],[MP5],[MP6])

We studied in detail the proof methodologies for verifying temporal properties of concurrent programs. Corresponding to the main classification of temporal properties into the classes of *safety* and *liveness* properties, appropriate proof principles were presented for each of the classes.

We developed proof principles for the establishment of *safety* properties. We showed that essentially there is only one such principle for safety proofs, the invariance principle, which is a generalization of the method of intermediate assertions. We also indicated special cases under which these assertions can be found algorithmically.

The proof principle that we developed for *liveness* properties is based on the notion of well-founded descent of ranking functions. However, because of the nondeterminancy inherent in concurrent computations, the well-founded principle must be modified in a way that is strongly dependent on the notion of *fairness* that is assumed in the computation. Consequently, three versions of the well-founded principle were presented, each corresponding to a different definition of fairness.

We illustrate the application of these rules on several examples. We suggest concise presentations of complex proofs using the devices of *transition tables* and *proof diagrams*.

- **Formal Approaches to the Construction of Correct Reactive Programs** ([MP7])

Reactive programs are programs whose role is to maintain an on-going interaction with their environment, rather than to produce a computational final result on termination. Programs belonging to this class are usually described by some of the attributes: concurrent, distributed, real-time, and typical examples of such programs are: embedded programs, communication networks, control programs of industrial plants, operating systems, real-time programs, etc. Clearly, the correct and reliable construction of such programs is one of the most challenging programming tasks existing today.

Due to the special character of these programs, the formal approach to their specification and development must be based on the study of their *behavior*, rather than on the function or relation they compute, an approach which is adequate for computational and sequential programs.

We developed formal methods for the specification, verification and development of reactive programs, based on the formalism of *temporal logic* that has been specifically developed to reason about behaviors of reactive programs.

Among the topics we investigated are:

- Modeling reactive programs as fair transition systems.
- The language of temporal logic and its usage for the specification of program properties and system requirements.
- A classification of specifications into the classes of *safety* properties, *responsiveness* properties, etc.
- Methodologies for formal verification of the safety properties of a reactive program, and development approaches derived from them.
- Methodologies for formal verification of the responsiveness properties of a reactive program, and derived development approaches.
- The case of finite-state programs and automatic verification tools for this case.

- **Tools and Rules for the Practicing Verifier** ([MP3],[MP4])

We developed a minimal proof theory which is adequate for proving the main important temporal properties of reactive programs. The properties we consider consist of the classes of *invariance*, *response*, and *precedence* properties. For each of these classes we present a small set of rules that is complete for verifying properties belonging to this class. We illustrate the application of these rules by analyzing and verifying the properties of a new algorithm for mutual exclusion.

AUTOMATED DEDUCTION

- **TABLEAU, An Interactive Graphic Deductive System** ([BMW])

We have collaborated with a team in developing an interactive theorem prover, TABLEAU, based on the deductive-tableau framework. Implemented on an Apple Macintosh computer, the system exploits the graphical interface of that machine in communicating with the user. Rather than relying on the keyboard, the user may select with a mouse which step in the proof to attempt next. Although directing the proof is the responsibility of the user, the system intervenes if the user attempts an illegal step. The system can construct proofs in propositional and predicate logic, in theories of the nonnegative integers, trees, and tuples, and in new theories introduced by the user.

The system has a combination of features lacking elsewhere. In particular,

- It can handle theorems with both universal and existential quantifiers.
- It can produce proofs by mathematical induction, including well-founded induction.
- It has special provisions for reasoning about equality. Furthermore, the convenient interface of the system makes it far easier to use in constructing a detailed proof than it is to prove the same result by hand, a feature unfortunately not shared with many systems.

The system has been augumented in several directions:

- Introduce the capability of adding new deduction rules. This would facilitate the application of the system to new theories.
- Introduce a facility for extracting information from proofs. This information could be a program, a plan, or a database transaction.
- Allow the gradual automation of the system. In particular, we would like to automate certain routine and repetitive aspects of the proof process.

We are also in the process of developing an interactive program synthesis system, based on our deductive techniques for program synthesis.

- **A Resolution Approach to Temporal Proofs** ([AM2])

A novel proof system for temporal logic was developed. The system is based on the classical non-clausal resolution method, and involves a special treatment of quantifiers and temporal operators.

Soundness and completeness issues of resolution and other related systems were investigated. While no effective proof method for temporal logic can be complete, we established that a simple extension of the resolution system is as powerful as Peano Arithmetic.

We have investigated the use of the system for verifying concurrent programs. We also provided analogous resolution systems for other useful modal logics, such as certain modal logics of knowledge and belief.

- **Logic: The Calculus of Computer Science** ([MW2])

The research papers in which we have presented the deductive approach to program synthesis has been addressed to the usual academic readers of the scholarly journals. In an effort to make this work accessible to a wider audience, including computer science undergraduates and programmers, we have developed a more elementary treatment in the form of a book, *The Logical Basis for Computer Programming*, Addison-Wesley.

This book requires no computer programming and no mathematics other than an intuitive understanding of sets, relations, functions, and numbers; the level of exposition is elementary. Nevertheless, the text presents some novel research results, including

- theories of strings, trees, lists, finite sets and bags, which are particularly well suited to theorem-proving and program-synthesis applications;
- formalizations of parsing, infinite sequences, expressions, substitutions, and unification;
- a nonclausal version of skolemization;
- a treatment of mathematical induction in the deductive-tableau framework.

DEDUCTIVE SYNTHESIS

- **The Deductive Synthesis of Computer Programs** ([MW1])

Program synthesis is the systematic derivation of a computer program to meet a given specification. Here the specification is a general description of the purpose of the desired program, while the program is a detailed description of a method for achieving that purpose. The particular emphasis of this project is on the development of deductive techniques, i.e., techniques based on theorem proving, for program synthesis. These techniques are amenable to automatic implementation, but may be used interactively or for the precise communication of derivations discovered by hand.

Some achievements of the project are as follows:

- Synthesis of a class of recursive unification algorithms (algorithms for finding a common instance of two expressions).
- Development of a situational logic for the synthesis of nonapplicative programs (programs with side effects).
- Introduction of deduction rules giving accelerated performance for relations of special importance to program synthesis (such as equality and ordering relations).
- Synthesis of a class of real-number algorithms employing the binary-search technique (such as binary-search square-root algorithm).
- Completion of a survey summarizing on a popular level the deductive approach to program synthesis.

REAL-TIME SYSTEMS

- **Real-Time Reasoning in Temporal Logic** ([AH1])

We introduce a real-time temporal logic for the specification of reactive systems. The novel feature of our logic, TPTL (Time Propositional Temporal Logic), is the adoption of temporal operators as quantifiers over time variables: every modality binds a variable to the time(s) it refers to.

TPTL is demonstrated to be both a natural specification language as well as a suitable formalism for verification and synthesis. We present a tableau-based decision procedure and model-checking algorithm for TPTL. Several generalizations of TPTL are shown to be highly undecidable.

- **Real-time Logics: Complexity and Expressiveness** ([AH2])

The theory of the natural numbers with linear order and monadic predicates underlies propositional linear temporal logic. To study temporal logics for real-time systems, we combine this classical theory of infinite state sequences with a theory of time, via a monotonic function that maps every state to its time. The resulting theory of timed state sequences is shown to be decidable, albeit nonelementary, and its expressive power is characterized by Omega-regular sets. Several more expressive variants are proved to be highly undecidable.

This framework allows us to classify a wide variety of real-time logics according to their complexity and expressiveness. In fact, it follows that most formalisms proposed in the literature cannot be decided. We are, however, able to identify two elementary real-time temporal logics as expressively complete fragments of the theory of timed state sequences, and give tableau-based decision procedures. Consequently, these two formalisms are well-suited for the specification and verification of real-time systems.

- **Proving Properties of Real-Time Systems** ([H1])

We introduced a novel extension of propositional modal logic that is interpreted over Kripke structures in which a value is associated with every possible world. These values are, however, not treated as full first-order objects; they can be accessed only by a very restricted form of quantification: the "*freeze*" quantifier binds a variable to the value of the current world. We present a complete proof system for this ("*half-order*") modal logic.

As a special case, we obtain the real-time temporal logic TPTL of [AH1]: The models are restricted to infinite sequences of states, whose values are monotonically increasing natural numbers. The ordering relation between states is interpreted as temporal precedence, while the value associated with a state is interpreted as its "real" time. We extend our proof system to be complete for TPTL, and demonstrate how it can be used to derive real-time properties of reactive systems.

Other applications of the *freeze* quantifier are in dynamic logic and epistemic logics.

- **Temporal Proof Methodologies for Real-time Systems** ([HMP])

We extend the specification language of temporal logic, the verification framework, and the underlying computational model to deal with real-time properties of concurrent and reactive systems. A global, discrete, and asynchronous clock is incorporated into the model by defining the abstract notion of a real-time transition system as a conservative extension of traditional transition systems: qualitative fairness requirements are replaced (and superseded) by quantitative lower-bound and upper-bound real-time requirements for transitions.

We show how to model programs that communicate either through shared variables or by message passing, and how to represent the important real-time constructs of priorities (interrupts), scheduling, and timeouts in this framework.

Two styles for the specification of real-time properties are presented. The first style uses bounded versions of the temporal operators: the real-time requirements expressed in this style are classified into *bounded-invariance* and *bounded-response* properties. The second specification style allows explicit references to the current time through a special clock variable.

Corresponding to the two styles of specification, we present two very different proof methodologies for the verification of real-time properties expressed in these styles. For the *bounded-operators* style, we provide proof rules to establish lower and upper real-time bounds for response properties of real-time transition systems. For the ***explicit-clock*** style, we exploit the observation that when given access to the clock, every real-time property can be reformulated as a safety property, and use the standard temporal rules for establishing safety properties.

THEORY OF PROGRAMMING

- **Logic Programming Semantics: Techniques and Applications** ([B1]-[B3])

It is generally agreed that providing a precise formal semantics for a programming language is helpful in fully understanding the language. This is especially true in the case of logic-programming-like languages for which the underlying logic provides a well-defined but insufficient semantic basis. Indeed, in addition to the usual model-theoretic semantics of the logic, proof-theoretic deduction plays a crucial role in understanding logic programs. Moreover, for specific implementations of logic programming, e.g. PROLOG, the notion of deduction stategy is also important.

We provided semantics for two types of logic programming languages and develop applications of these semantics. First, we propose a semantics of PROLOG programs that we use as the basis of a proof method for termination properties of PROLOG programs. Second, we turn to the temporal logic programming language TEMPLOG of Abadi and Manna, develop its declarative semantics, and then use this semantics to prove a completeness result for a fragment of temporal logic and to study TEMPLOG's expressiveness.

In our PROLOG semantics, a program is viewed as a function mapping a goal to a finite or infinite sequence of answer substitutions. The meaning of a program is then given by the least solution of a system of functional equations associated with the program. These equations are taken as axioms in a first-order theory in which various program properties, especially termination or non-termination properties, can be proved. The method extends to PROLOG programs with extra-logical features such as ***cut***.

For TEMPLOG, we provide two equivalent formulations of the declarative semantics: in terms of a minimal temporal Herbrand model and in terms of a least fixpoint. Using the least fixpoint semantics, we are able to prove that TEMPLOG is a fragment of temporal logic that admits a complete proof system. This semantics also enables us to study TEMPLOG's expressiveness. For this, we focus on the propositional fragment of TEMPLOG and prove that the expressiveness of propositional TEMPLOG queries essentially corresponds to that of finite automata.

- **Temporal Logic Programming** ([AM1])

Temporal logic is a formalism for reasoning about a changing world. Because the concept of time is directly built into the formalism, temporal logic has been widely used as a specification language for programs where the notion of time is central. For the same reason, it is natural to write such programs directly in temporal logic. We developed a temporal logic programming language, TEMPLOG, which extends classical logic programming languages, such as PROLOG, to include programs with temporal constructs. A PROLOG program is a collection of classical *Horn clauses*. A TEMPLOG program is a collection of *temporal Horn clauses*, that is, Horn clauses with certain temporal operators. An efficient interpreter for PROLOG is based on *SLD-resolution*. We base an interpreter for TEMPLOG on a restricted form of our temporal resolution system, *temporal SLD-resolution*.

- **A Hierarchy of Temporal Properties** ([MP1])

We proposed a classification of temporal properties into a hierarchy which refines the known *safety-liveness* classification of properties. The classification is based on the different ways a property of finite computations can be extended into a property of infinite computations.

This hierarchy was studied from three different perspectives, which were shown to agree. Respectively, we examined the cases in which the finitary properties, and the infinitary properties extending them, are unrestricted, specifable by temporal logic, and specifiable by predicate automata. The unrestricted view leads also to a topological characterization of the hierarchy as occupying the lowest two levels in the Borel hierarchy.

For properties that are expressible by temporal logic and predicate automata, we provide a syntactic characterization of the formulae and automata that specify properties of the different classes. The temporal logic characterization strongly relies on the use of the past temporal operators.

Corresponding to each class of properties, we presented a proof principle that is adequate for proving the validity of properties in that class.

PH.D. THESES

- **The Temporal Specifications and Verifications of Real-Time Systems** ([H2])

We generalize the temporal-logic approach to the analysis of reactive systems to real-time systems. The complete methodology includes the following four components:

- an abstract *computational model* for real-time systems. We add a global, discrete, and asynchronous clock to the interleaving model of concurrency. Traditional transition systems are extended to real-time transition systems by including lower-bound and upper-bound real-time constraints for transitions. We show how to model both communication by shared variables and by message passing.
- a formal *specification language*. We analyze and compare the complexity and expressiveness of three extensions of temporal logic. GCTL refers to time through a special clock variable: TPTL replaces the clock variable by the "freeze" quantifier; MTL expresses timing constraints by bounded temporal operators. It is shown that while all three languages are equally expressive, only the latter two logics have elementary decision problems.
- a *model-checking algorithm* for the verification of finite-state systems. We give tableau-decision procedures for both TPTL and MTL, and show how these algorithms can be used to verify properties of finite-state real-time transition systems. The automatic-verification problem is shown to be unsolvable for more expressive specification languages.
- a (relative) complete *proof system*. The proof system for the deductive verification of real-time transition systems consists of two parts. In the general part, we give a complete proof system for TPTL. In the program part, we present relative complete proof rules for the verification

of bounded-invariance and bounded response properties of a given real-time transition system. Two distinct real-time verification styles are contrasted.

- **Temporal Reasoning for Real-Time Systems** ([A])

Various temporal logics and ω-automata have been proposed and extensively studied for formal specification and reasoning about the behavior of reactive systems over time. However, most of these methods are for qualitative temporal reasoning: there is a large class of systems, called *real-time systems* — systems whose correctness depends on the actual timing of events, that cannot be handled by these methods. For example, the standard temporal logics cannot specify the hard real-time requirements such as, the system responds within 5 seconds. Also, while modeling a system, there is no way to model the timing delays and constraints of the physical components. With the increasing use of the computers for real-time applications, there is a pressing need for developing formal methods for the analysis of time-dependent systems. In this thesis, we extend the scope of the temporal logic and automata based methods to the real-time systems.

We consider ways to generalize the syntax and the semantics of the temporal logics so as to allow expression of the real-time constraints. We address questions such as what is the right notation, how to model time, and what type of timing constraints should be allowed. We study different theoretical issues such as complexity, and expressiveness, and whenever possible, give decision procedures.

A particularly interesting case is when time is modeled by a dense linear order. In this framework, we consider how to model a real-time system as a finite-state automaton (augmented with a mechanism to model delays). We formalize the real-time equivalent of the notion of regular trace sets, and study its language-theoretic properties useful in verification. We also develop model-checking algorithms based on our model.

We believe that the results of this thesis provide a basis for the future attempts towards the formal verification of hardware, communication protocols, control systems, and asynchronous concurrent systems in general.

PUBLICATIONS

Research papers and Ph.D. theses partially supported by this contract.

INVITED PAPERS (CONFERENCES)

[MP1] Z. Manna and A. Pnueli. "A hierarchy of temporal properties," *9th Symposium on Principles of Distributed Computing*, Québec, Canada, Aug. 1990.

[MP2] Z. Manna and A. Pnueli. "A temporal proof methodology for reactive systems," *5th Jerusalem Conference on Information Technology*, Jerusalem, Oct. 1990, pp. 757–773.

INVITED PAPERS (SPECIAL EVENTS)

[MP3] Z. Manna and A. Pnueli. "An exercise in the verification of multi-process programs" in *Beauty is our business* (W.H.J. Feijen, A.J.M. Van Gasteren, D. Gries, J. Misra, eds.), Springer-Verlag, New York, 1990, pp. 289–301.

[MP4] Z. Manna and A. Pnueli. "Tools and rules for the practicing verifier," *Carnegie Mellon computer Science: A 25-year Commemorative*, ACM Press and Addison–Wesley, September 1990.

[MP5] Z. Manna and A. Pnueli. "The anchored version of the temporal framework," in *Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency* (J.W. de Bakker, W.P. de Roever, and G. Rozenberg, eds.), Lecture Notes in Computer Science 354, Springer-Verlag, 1989, pp. 201–284.

[MW1] Z. Manna and R. Waldinger, "Fundamentals of Deductive Program Synthesis," in *Logic, Algebra, and Computation*, NATO ASI Series, Springer-Verlag, 1991.

JOURNAL ARTICLES

[AM1] M. Abadi and Z. Manna. "Temporal logic programming", Journal of Symbolic Computation, Vol. 8, No. 3 (Sept. 1989), pp. 277–295.

[AM2] M. Abadi and Z. Manna, "Nonclausal deduction in first-order temporal logic", Journal of the ACM, Vol. 37, No. 2 (April 1990), pp. 279–317.

[MP6] Z. Manna and A. Pnueli. "Completing the temporal picture", Theoretical Computer Science Journal (in press).

PAPERS BY PH.D. STUDENTS

Supervised by Zohar Manna

[AH1] R. Alur and T.A. Henzinger. "A Really Temporal Logic", 13th IEEE Symposium on Foundations of Computer Science, 1989, pp. 164–169.

[AH2] R. Alur and T.A. Henzinger. "Real-time Logics: Complexity and Expressiveness", Symposium on Logic in Computer Science, Philadelphia, PA, June 1990.

[B1] M. Baudinet, "Proving Termination Properties of PROLOG Programs: A Semantic Approach," *3rd Annual Symposium on Logic in Computer Science*, pp. 336–347, Edinburgh, Scotland, July 1988.

[B2] M. Baudinet, "Temporal Logic Programming is Complete and Expressive," *6th ACM Symposium on Principles of Programming Languages*, Austin, Texas, January 1989.

[HMP] T. Henzinger, Z. Manna, and A. Pnueli, "An interleaving model for real time," *5th Jerusalem Conference on Information Technology*, October 1990.

[H1] T. Henzinger, "Half-order modal logic: How to prove real-time properties," *9th Symposium on Principle of Distributed Computing*, Québec, Canada, August 1990.

SOFTWARE

Project Supervised by Zohar Manna and Richard Waldinger

[BMW] R. Burback, Z. Manna, S. A. Fraser, H. McGuire, J. Smith, R. Waldinger, and M. Winstandley, *Tableau Deductive System*, in MacIntosh Educational Software Collection, distributed by Chariot Software Group, San Diego, 1990. (The User Manual, "Using the Deductive Tableau System," is enclosed.)

Ph.D. THESES

Supervised by Zohar Manna

[A] R. Alur, "Temporal Reasoning for Real-Time Systems," Ph.D. Thesis, Computer Science Department, Stanford University (to appear, 1991).

[B3] M. Baudinet, "Logic Programming Semantics: Techniques and Applications," Ph.D. Thesis, Computer Science Department, Stanford University (1989).

[H2] T. Henzinger, "The Logic of Real-Time Systems," Ph.D. Thesis, Computer Science Department, Stanford University (to appear, 1991).

TEXTBOOKS

The research behind these books was partially supported by this contract.

[MW2] Z. Manna and R. Waldinger, *Logical Basis for Computer Programming*, Addison-Wesley Pub., Reading, MA.
Vol. 2: Deductive Systems (January 1990).
The Concise Version (to appear, 1991).

[MP7] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive Programs*, Springer-Verlag, NY (to appear, 1991).